Efficient Polyline Surface Mapping with Strong Error Bounds

Yujie Lian[1,*], Georg Rempfer[2,*], Hadi Askaripoor[1], Jonas Landsgesell[2] ,and Alois Knoll[1]

IECON 2021 - 47th Annual Conference of the IEEE Industrial Electronics Society | 978-1-6654-3554-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/IECON48115.2021.9589795

***Abstract*— We propose a novel surface mapping algorithm processing sensor data from, *e.g.*, stereo or structure from motion video, triangulating ultrasonic, lidar, or tactile sensors. The algorithm does not rely on a grid and instead constructs a set of polylines as an extremely efficient representation of surrounding surfaces. We use ridges of a continuous occupancy to extract surfaces from location measurements which are subject to noise. Our algorithm for iterative map updates is efficient, stable, unconditionally convergent, gives strong guarantees about the map accuracy, and is invariant under permutation of the input data. It does not suffer from association errors, and operations such as shape updates, splitting, and merging of surfaces arise naturally and are not handled separately. The algorithm corrects wrongly mapped surfaces with future more accurate measurements, and the polyline approximation of the ridge can be tuned to arbitrary accuracy. Runtime measurements demonstrate the feasibility of the approach for embedded applications.**

I. Introduction

Many applications in autonomous movement of robots and path planning rely on accurate models of the robots environment. Maneuvering in tight spaces occupied by irregularly shaped objects such as warehouses, production facilities, and private apartments requires information about the shapes and locations of surrounding objects on a centimeter scale. At the same time, these functions are often implemented on low-cost embedded computing platforms. The limited memory and computational power these platforms provide make existing environment modeling algorithms based on grid maps [1] unsuitable for this application as the accuracy of the environment model is limited by the available memory and calculation time.

One approach to circumvent the trade-off between mapping accuracy, size, and computational resources on embedded platforms is to employ an environment model that only maps relevant surfaces instead of the whole volume [2], [3]. Such a model makes more efficient use of the available resources for multiple reasons: First, empty spaces containing no surfaces don't need to be modeled at all; second, surfaces are of lower dimension than the volume and require less information to represent; and third, discretization of the surfaces by means of polygons lends itself naturally to adaptive resolution schemes, further reducing the computational cost.

A downside of this class of algorithms is that the mathematical modeling of consistent update schemes is challenging. While it is straight-forward to apply update schemes based e.g. on Bayes' rule to grid-based models, thereby gaining important properties like permutation invariance with respect to the order of the input data, the same is much harder to achieve for surface mapping algorithms. Here new sensor data is either used to create new surfaces, or associated with existing ones and used to improve their shape and location accuracy. Since it is impossible in principle to distinguish perfectly between these two cases, surface mapping algorithms suffer from association errors: if an initially created surface is accurate, the algorithm performs excellently – outliers are detected as such and accurate measurements are used to further improve the surface model. If, however, the initially created surface was inaccurate due to outliers appearing early in its creation, the algorithm rejects future accurate measurements and potentially uses future outliers to further worsen the accuracy of the map.

For specific applications, these misassociations and their consequences might be reduced to acceptable levels by using heuristics and short measurement buffers that allow to correct for wrong associations. However, to solve this problem in general and in a way that can be implemented on an embedded platform, a novel algorithm is needed. In this investigation, we aim to produce a proof-of-concept implementation of a mapping algorithm that represents the environment as a set of surfaces discretized as polygons. The algorithm we propose does not rely on a grid but nevertheless preserves permutation symmetry of the input data, avoiding the aforementioned association problem by design. Operations like extending, merging, splitting, and updating of surfaces arise naturally from the formalism and don't require separate handling.

We quantify the accuracy of maps resulting from our approach and discuss systematic errors in detail. A particular focus of our proof-of-concept is computational efficiency. We demonstrate how a combination of fast neighbor search and an iterative update scheme makes our approach feasible to be implemented on recent embedded platforms for real-time mapping.

II. Related Work

Mapping buildings is one of the most important tasks in perception and navigation of mobile robots. Elfes [1] proposed occupancy grids for map generation and representation, which has been widely applied in robot mapping problems. Occupancy grids divide the space into cells, each of which carries am occupancy probability. However, this approach suffers two major drawbacks: independence between cells and a necessary trade-off between map resolution, size, and memory consumption.

*The first two authors have equal contribution, the order just reflects alphabetical order.

[1]Technical University of Munich, Germany
[2]Robert Bosch GmbH, Germany

To overcome the major drawbacks of the occupancy grid maps, O'Callaghan *et al.* [4] proposed an approach for contextual occupancy maps in a continuous manner using Gaussian processes. The previously estimated surfaces can then be updated using Bayes' rule with the new observations. However, Gaussian processes require hyperparameters that need to be learned which leads to difficulties in applying this approach to an online mapping application in real-time.

Early approaches representing the environment as a set of surfaces used sets of straight lines of infinite extent. Sack *et al.* [5] give a review about algorithms making use of this representation. The main drawbacks of this class of algorithm are that they lack the ability to represent convex structures, and that the resulting maps don't contain information about the connectivity of surfaces. Later developments focus on the use of polylines to overcome these limitations. Navarro *et al.* [6] demonstrate this approach for online mapping using an infrared sensor ring. Among all the polyline extraction methods, only a few are based on a rigorous mathematical framework. One such method by Pfister *et al.* [7] constructs a line-based map that best fits 2D range readings using a maximum likelihood formalism. Schaefer *et al.* [8] derive a consistent algorithm for polyline extraction from 2D laser row range scans. The core idea is to produce a polyline representation of the environment that describes the maximum likelihood environment compatible with the observed measurements. Ozertem *et al.* [9] propose a method referred to as subspace constrained mean shift (SCMS) to construct polylines as low-dimensional representations for clusters of points in one and two dimensions. They extend the mean shift method [10], arriving at the same formalism of principal curves of a kernel density estimation that we employ in this investigation. We add to their work by proposing a computationally efficient method for constructing and updating the polylines such that they obey strict error bounds.

III. Surfaces as Ridges

Our approach defines surfaces in terms of ridges (also called principal curves) of a continuous occupancy. The ridges are derived from randomly distributed measurements of static surfaces.

In order to estimate the true position of a surface, we associate each measurement with a Gaussian smoothing kernel. The superposition of the smoothing kernels has a ridge which depends on the random realization of measurements. Our algorithm efficiently computes a discretized version of this ridge which we refer to as a polyline. While the ridge itself is subject to random fluctuations arising from the stochastic nature of the measurements, the polyline representation of the ridge obeys strictly controllable error bounds.

In the following, we will define the necessary concepts and discuss the approach in detail.

A. Continuous Occupancy

As input, we expect measurement locations at positions $\mu_i \in \mathbb{R}^2$ in a fixed coordinate system. To derive a continuous occupancy, we associate a two-dimensional normal distribution $P_i(x)$ as smoothing kernel to each measurement i.

$$P_i(x) = \frac{1}{2\pi\sqrt{|S|}} \exp\left(-\frac{1}{2}\langle S^{-1}(x-\mu_i),(x-\mu_i)\rangle\right) \quad (1)$$

In the following, the covariance matrix S is a diagonal matrix with entries σ^2. We do not assign any probabilistic interpretation to these normal distributions, but rather just use them as smoothing kernels and call the distance σ the *smoothing parameter*.

The superposition of the normal distributions of all available measurements forms the continuous occupancy $L(x)$, whose value, gradient, and Hessian read

$$L(x) = \sum_i P_i(x), \quad (2)$$

$$\nabla L(x) = \sum_i -P_i(x)(S^{-1}(x-\mu_i)), \quad (3)$$

$$HL(x) = \sum_i P_i(x)\{ (S^{-1}(x-\mu_i)) \otimes (S^{-1}(x-\mu_i)) - S^{-1}\}. \quad (4)$$

Fig. 1 (left) depicts the continuous occupancy for a sample set of measurements.

B. Ridge Criterion

We are interested in the ridges or principal curves of the continuous occupancy. A point is part a ridge if it is a maximum along the direction of smallest principal curvature of the continous occupancy. Using the gradient ∇L and Hessian HL of the occupancy, the ridge criterion can be expressed in the following way, similar to the definition in [9]:

$$HL(x)v_1(x) = \lambda_1(x)v_1(x) \quad (5)$$

$$\lambda_1(x) < 0 \quad (6)$$

$$\langle \nabla L(x), v_1(x)\rangle = 0 \quad (7)$$

Eq. (5) defines the eigenvector v_1 of the Hessian HL. We assume the eigenvalue λ_k to be associated with the eigenvector v_k, with λ_1 being the smaller of the two eigenvalues. The eigenvectors point along the directions of principal curvature of the continuous occupancy L. $\lambda_1(x) < 0$ requires that the curvature in this direction to be negative, *i.e.*, x be on a ridge rather than in a valley. The third Eq. (7) demands that there is slope only along the direction perpendicular to v_1. Therefore, v_1 forms the aforementioned direction along which x is a maximum. x would form a locally extremal point if x were also a maximum along the direction perpendicular to v_1. This is possible as there can be peaks and saddle points along a ridge.

We define the ridge function by taking the absolute of Eq. (7)

$$R(x) := |\langle \nabla L(x), v_1(x)\rangle| \stackrel{!}{=} 0, \quad (8)$$

which removes the ambiguity of the sign of the ridge criterion due to the ambiguity of the orientation of the eigenvector $v_1(x)$ simplifying the numerical treatment. Fig. 1 (center)

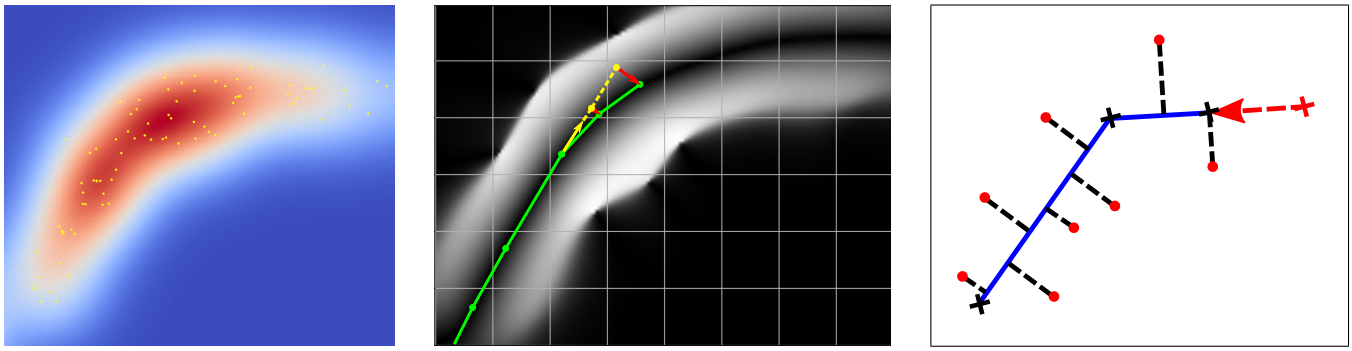

Fig. 1. **Left:** Continuous occupancy L (blue-white-red background) resulting from a set of noisy measurements (yellow dots). **Center:** The ridge function R from Eq. (8) (black-white background) for the same set of measurements. The black line represents the zero-set of the ridge function – the ridge. Also shown is a polyline (green) during construction, the new candidate node (yellow end node) is placed one step-size in the from the previous end node in the direction of $v_2(x)$ (yellow arrow). It undergoes relaxation (red arrow) with a relaxation distance that is large enough to warrant insertion of a middle node (also yellow). The middle node undergoes relaxation onto the ridge as well, and both nodes become part of the polyline approximation of the ridge. These few polyline segments demonstrate how the algorithm increases the segment size on straight parts of the ridge, and decrease the step-size on curved segments to maintain a global upper bound for the approximation error. **Right:** Measurements (red) projected onto the segments of the polyline (blue) to determine where to terminate the polyline, and where to optimally place the end node to avoid unwarranted extension.

depicts the ridge function, as well as its zero-set, the ridge, for a sample continuous occupancy.

IV. Constructing and Updating the Ridge Map

To take advantage of the local nature of the mapping problem – measurements only influence mapped surfaces close to their location – we employ an update scheme that leaves most of the existing ridge map intact, only updating it where necessary. We first cluster measurements received in one cycle whose bounding box defines an update region. Subsequently, we apply the ridge construction scheme described in detail in Sec. V to each of the update regions. Finally, we replace those parts of the polylines of the global map that intersect an update region with the respective newly constructed ones, making sure to connect them properly at the respective update region boundary intersections.

V. Constructing an Individual Ridge

In the following, we describe a computationally efficient trust-region method to approximate ridges by means of polylines with adaptively chosen segment lengths. The algorithm respects hard error bounds as we will show in the following. These hard error bounds and the fact that we can tune them to arbitrarily small values will allow us to quantify systematic errors of the ridge map in Sec. VI-C.

A. Initial Guesses

During most update steps, the existing ridges will just be slightly perturbed by new measurements. Therefore, the intersections of ridges in the current map with the update region's boundary form excellent initial guesses. Not all ridges in the update region necessarily already existed before the current update. For those that didn't, the local maxima of the continuous occupancy form good initial guesses. We identify those local maxima using a two-dimensional trust-region Newton optimization scheme with any measurement not close to an existing polyline as starting point. Since ridges only exist in regions with measurements, we can guarantee that all ridges are constructed.

B. Trust-Region Ridge Extension

Given a starting point on a ridge, we efficiently construct a polyline approximating the complete ridge using a variant of the trust-region method adapted to our specific problem. Fig. 1 (center) depicts all steps of the trust-region ridge extension.

1) Sub-problem of the Modified Trust-Region Method: We derive a local tangent to the ridge in terms of eigenvectors of the continuous occupancy. The eigenvector $v_1(x)$ associated with the smaller eigenvalue marks a direction perpendicular to the ridge as required by Eq. (7). Since the eigenvectors of the symmetric Hessian $HL(x)$ are perpendicular, the eigenvector $v_2(x)$ associated with the bigger eigenvalue must therefore be tangent to the ridge. A step along the search direction $v_2(x)$ results in a new candidate node extending the polyline along the ridge.

2) Candidate Ridge Node Relaxation: However, since the ridge in general, has non-zero curvature, the new candidate node will be off the ridge – by how much is determined by the trust-region step size and the ridge curvature.

Leaving this error uncorrected is not an option, as it would propagate into all nodes further along the ridge, rendering its polyline approximation progressively worse. To maintain strong error bounds, we relax the candidate node back onto the ridge. We do this by applying Newton's method for root-finding to the ridge function. This relaxation only needs to be done in the direction perpendicular to the last polyline segment, as it forms a good approximation to the search direction given by a full two-dimensional Newton update. A suitable step size based termination criterion allows us to control the accuracy-runtime trade-off of this step arbitrarily.

We use the distance of the correction during this relaxation $d_{\text{relaxation}}$ as a quality indicator for the linear approximation of the ridge during the trust-region step size update.

3) Step-Size Update: We implement the step-size control based on the quality indicator $d_{\text{relaxation}}$ in a similar manner to the original trust-region method [11]: If it takes a large

Fig. 2. A complex scene consisting of multiple corners and straight walls mimicking a typical indoor environment consisting of a 6 m wide hallway. **Left:** The robot in the simulation moves along a the hallway and around a 90° corner, producing measurements (green dots) of the surrounding surfaces while moving (red trajectory). **Center:** The continuous occupancy and ridge function (black to white background) are created and updated on the fly. **Right:** The resulting ridge map (red and blue polylines). The zoom-in shows systematic corner-cutting.

$d_{\text{relaxation}} > 0.75\sigma$ to relax back onto the ridge, then this indicates that the step size is too large for the local curvature and the step-size for the next step is halved. If $d_{\text{relaxation}} < 0.25\sigma$, then we double the step-size. Otherwise, the step size remains unchanged. Choosing the step-size adaptively not only ensures that all of our approximations remain valid, but also results in polylines with variable segment sizes that adapt to the ridges' local curvature. Furthermore, we can again control the accuracy-runtime trade-off of the iteration along the ridge arbitrarily by influencing the step-sizes chosen.

4) Middle Node Insertion: To be able to give accuracy guarantees not only of the ridge nodes which underwent relaxation, but also every other point on the segments between these nodes, we need to control the on-segment error as well. We assume that the on-segment of a new segment is smaller than its farthest extrapolated point – the new candidate node. So special measures are only necessary when the candidate node overshoots but the relaxation corrects the candidate node's error. If the correction exceeds the error bound requiring a step-size decrease, we insert a middle node into the new segment and relax that one back onto the ridge as well.

C. Terminating a Ridge

The termination criterion consists of two conditions. Firstly, we require that $L(x)$ exceed a minimum threshold L_{tol} which is only reached if the desired number of measurements' distributions overlap to filter outliers. Secondly, the algorithm terminates when there are no measurements close to the candidate node. We utilize nearest neighbor search to find all measurements near the ridge and project them onto the ridge segments to eliminate the distance contribution from measurement error in the direction perpendicular to the ridge (since this is a property of the sensor and not of the surface geometry). We calculate the distance $d_{\text{projected}}$ between the closest projected measurement and the candidate ridge node. If this distance is larger than a given threshold d_{tol}, the ridge will be terminated. To prevent overshoot of the last polyline node, we adjust its position in such a way that the polyline's end node position coincides with the projected position of the last measurement as shown in Fig. 1 (right).

This processing step completes the core algorithm for the efficient construction of polylines as discrete representations of ridges of the continuous occupancy.

VI. Results

As a metric for the accuracy of the polylines created by our algorithm, we use the root-mean-square deviation (RMSD) to measure the difference between the generated polylines and ground truth surfaces from which sets of synthetic measurements were created. The RMSD evaluates the Euclidean distance between a point on a polyline and the closest point on the ground truth surface. For polylines like the ones we use, the RMSD can be expressed as:

$$\text{RMSD} = \sqrt{\frac{1}{\sum_i l_i} \sum_i l_i \int_0^1 d_{\min}\left(x_i + t \cdot [x_{i+1} - x_i]\right)^2 dt} \tag{9}$$

Here x_i and x_{i+1} denote the beginning and end points of the segments indexed by i that form the polyline and have a length $l_i = |x_{i+1} - x_i|$. The RMSD for a whole map is calculated as the average of all ridges in the map weighted with their respective lengths.

A. A Typical Indoor Environment as Benchmark Scene

In this section, the algorithm is applied to a complex scene that consists of multiple corners and straight walls, as shown in Fig. 2. The robot in the simulation moves along a hallway with a 90° corner. We create synthetic sensor data for the scene by first determining which parts of the ground truth surfaces are within the field-of-view (4 m from the robot position) and unoccluded by other surfaces. We then choose j positions (with uniform probability distribution) along the visible surfaces where j is the Poisson distributed number of measurements per cycle. The number of measurements from a visible segment is proportional to the segment's cross-section from the sensor perspective. Finally, we add Gaussian noise to the j positions on the segment with a variance proportional to the distance of the position to the sensor, and larger in the depth direction than in the transversal

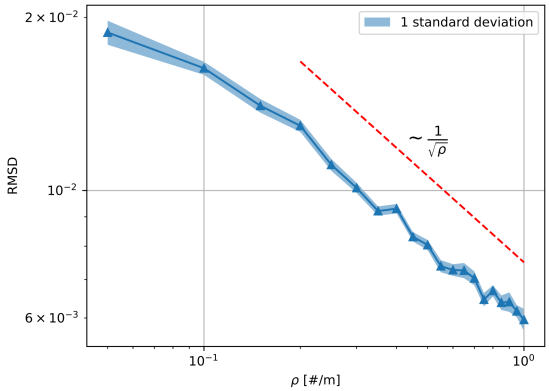

Fig. 3. Accuracy of the polyline approximating a straight wall from the complex scene in Fig. 2. Shown is the RMSD of the polyline for a range of different measurement densities (blue triangles). The RMSD values are mean values of 100 random sets of sensor data for every measurement density to eliminate the sampling error of the RMSD. The blue corridor marks a 1σ uncertainty interval for these mean values. The RMSD decreases as $\rho^{-0.5}$ with the measurement density, as is expected from theoretical arguments.

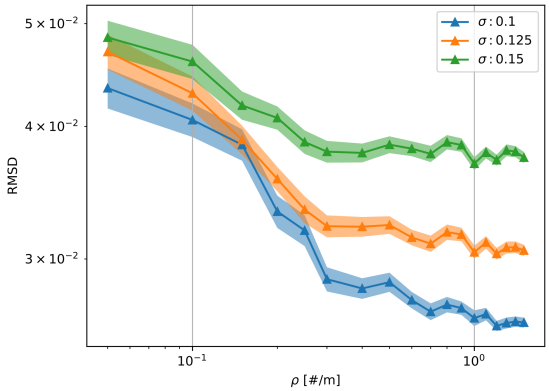

Fig. 4. Accuracy of the polyline approximating a corner from the complex scene in Fig. 2. The lines demonstrate how the RMSD depends on the measurement density for three different smoothing parameters. For small measurement densities, the RMSD drops with increasing measurement density for all smoothing parameters due to decreasing random errors similar to the straight surface in Fig. 3. At high measurement density, where random errors vanish, the RMSD becomes asymptotically constant due to systematic corner-cutting errors. These systematic errors decrease with smaller smoothing parameters and less measurement noise.

direction. A number of measurements are created for every position along the robot trajectory, a cumulative plot of which is shown in Fig. 2 (left). The ridge map update algorithm iteratively constructs the whole map as described in Sec. IV and shown in Fig. 2 (right). Fig. 3 and 4 depict the RMSD for straight and curved sections of the complex scene in Fig. 2, respectively. In the following, we investigate how the measurement density (determined by the sensor set and the speed of the robot), as well as the smoothing parameter, influence the accuracy of the resulting surface map.

B. Random Errors

The position of a point on the ridge can be understood as a weighted average of the locations of surrounding measurements (with weights deriving from the smoothing kernel) [9]. This implies that the ridge's root mean square deviation scales with $\rho^{-0.5}$, similar to how the standard deviation of the estimated mean scales like $N^{-0.5}$ where N denotes the sample size. Fig. 3 demonstrates that with our algorithm we do indeed create polylines that achieve this theoretical scaling for straight surfaces. This is only possible because our algorithm allows us to tune the thresholds for the maximum error of the polyline approximation of the ridge to arbitrarily small values. Here we tune these thresholds in favor of accuracy at the expense of calculation time to a level far below the sampling error of the ridge itself.

C. Systematic Errors due to Limited Curvature

As the zoom-in in Fig. 2 already shows, the ridges cut corners. Fig. 5 shows this systematic error in detail for low and high measurement density and negligible discretization errors of the polyline approximation. The left panel of Fig. 5 shows that at low measurement density, the systematic corner-cutting error is masked by the random sampling error. At high density where these random errors vanish, however, the systematic corner-cutting error dominates, as the right panel of Fig. 5 shows.

Evaluating the systematic corner mapping error reveals that while the ground truth surface assumes infinite curvature (and zero curvature radius) in the corner, the ridge reconstructed from the measurements assumes a finite maximum curvature with a radius of similar magnitude as the smoothing parameter used in the reconstruction and the measurement noise. The ridge is displaced to the inside of the corner due to an increased measurement density induced by the ground truth surface's curvature on the concave side of the surface. Even without any measurement noise, the effect would persist due to the smoothing applied to the measurements when deriving the continuous occupancy, albeit with a smaller magnitude. This systematic curvature induced mapping inaccuracy effectively sets a resolution limit for the surface reconstruction.

The RMSD of the ridge representing the corner shown in Fig. 4 demonstrates both of these facts: beyond a certain measurement density the corner-cutting errors dominate over the random mapping errors and the mapping accuracy does not improve any further; and the mapping accuracy achieved in this saturation regime improves with smaller smoothing parameter.

While these results suggest that small smoothing parameters are always optimal, this is not true in practice. In the low measurement density regime, errors that are not well represented by the RMSD of the ridges become significant, such as fragmenting of polylines with a large number of fragments of incorrect orientation. The measurement density of the environment is typically highly inhomogeneous since it depends on the variable speed of the robot as well as details of the surrounding surfaces. This suggests that an adaptive resolution scheme optimally choosing the smoothing parameter according to the local measurement density might be a useful extension to the algorithm.

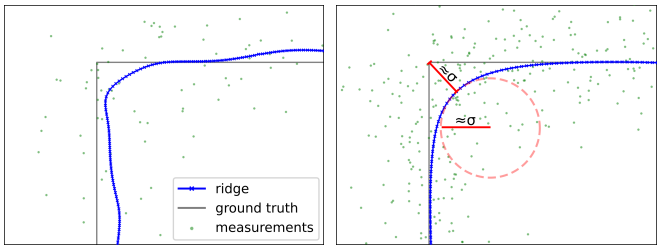

Fig. 5. Comparison of the ridges for a 90° corner with two different measurement densities. For the lower density measurements (left), the ridge suffers from both random mapping errors and systematic errors at the corner, while for the higher density measurement set (right), the random mapping errors decrease to a magnitude far below the systematic errors, which make up the major part of the inaccuracy. The measurement noise as well as the smoothing step in the construction of the continuous occupancy limits the maximum curvature of the ridges, which results in systematic corner-cutting behavior.

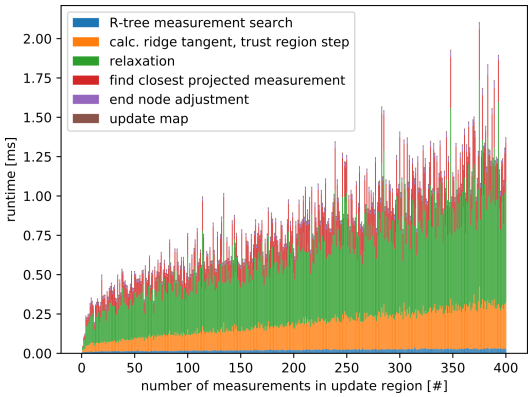

Fig. 6. Average runtime of a map update for one measurement cluster as a function of the number of measurements in the update region. The stacked plot also shows how the map update's runtime distributes over the different processing steps. Samples for this evaluation come from the simulations of the complex scene.

D. Computational Efficiency

We use the complex scene depicted in Fig. 2 for the runtime evaluations presented here. The environment which we use to run the algorithm is an Intel Xeon CPU E3-1505M v5 @ 2.80 GHz base / 3.70 GHz boost accessing 32 GB DDR4 ECC RAM @ 2133 MHz with a memory bandwidth of 34.1 GB/s (half of which is available to our sing-core implementation) running Windows 10 64-bit. The code implementing the algorithm is compiled using Visual Studio 2019 in 64-bit release mode with all optimizations enabled. We time the separate steps of the algorithm for every map update and show averages for map update with equal measurement count in Fig. 6. We observe $\mathcal{O}(n)$ scaling of the runtime with the number of measurements, which indicates that our efforts employing fast neighbor search to take advantage of the local nature of the update scheme were successful. If the number of measurements taken into account for the map update can be constrained to under 100, the runtime does not exceed 0.5 ms. Even for up to 400 measurements in the update region, the runtime typically stays below 1.25 ms with peaks of up to 2 ms. This computational cost is well within reach of current embedded systems running the proposed algorithm online and at high frequency.

Future research will focus on methods to eliminate redundant or irrelevant information from the measurement buffer to put hard limits in the number of measurements in an update region. This will allow us to give strong runtime guarantees.

VII. Conclusion

We gave a complete description of a formalism that reconstructs surfaces from discrete measurement locations in terms of ridges of a continuous occupancy. We also describe an algorithm to efficiently produce polyline approximations of these ridges which obey strict error bounds that can be tuned to arbitrary precision. The method does not employ a grid but the resulting map nevertheless is history independent, which allows the method to always correct initial mapping errors with future sensor data. In addition to that, operations such as shape updates, splitting and merging of polylines, and associations of measurements with polylines are not separately handled. They all arise as natural consequences of the formalism. We apply the method to a scene consisting of an indoor hallway and quantify the resulting map accuracy as a function of the measurement density and the smoothing parameter of the continuous occupancy. Timing results show that employing our implementation as part of an embedded system for online mapping is feasible.

References

[1] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, 1989.

[2] K. R. Laviers and G. L. Peterson, "Cognitive robot mapping with polylines and an absolute space representation," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, vol. 4. IEEE, 2004, pp. 3771–3776.

[3] M. A. Movafaghpour and E. Masehian, "Poly line map extraction in sensor-based mobile robot navigation using a consecutive clustering algorithm," *Robotics and Autonomous Systems*, vol. 60, no. 8, pp. 1078–1092, 2012.

[4] S. O'Callaghan, F. T. Ramos, and H. Durrant-Whyte, "Contextual occupancy maps using gaussian processes," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 1054–1060.

[5] D. Sack and W. Burgard, "A comparison of methods for line extraction from range data," in *Proc. of the 5th IFAC symposium on intelligent autonomous vehicles (IAV)*, vol. 33, 2004.

[6] D. Navarro, G. Benet, and F. Blanes, "Line-based incremental map building using infrared sensor ring," in *2008 IEEE International Conference on Emerging Technologies and Factory Automation*. IEEE, 2008, pp. 833–838.

[7] S. T. Pfister, S. I. Roumeliotis, and J. W. Burdick, "Weighted line fitting algorithms for mobile robot map building and efficient data representation," in *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, vol. 1. IEEE, 2003, pp. 1304–1311.

[8] A. Schaefer, D. Büscher, L. Luft, and W. Burgard, "A maximum likelihood approach to extract polylines from 2-d laser range scans," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4766–4773.

[9] U. Ozertem and D. Erdogmus, "Locally defined principal curves and surfaces," *The Journal of Machine Learning Research*, vol. 12, pp. 1249–1286, 2011.

[10] Yizong Cheng, "Mean shift, mode seeking, and clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 8, pp. 790–799, 1995.

[11] D. C. Sorensen, "Newtons method with a model trust region modification," *SIAM Journal on Numerical Analysis*, vol. 19, no. 2, pp. 409–426, 1982.